Discovering Dense Correlated Subgraphs in Dynamic Networks

Giulia Preti[1]([⊠]), Polina Rozenshtein[2], Aristides Gionis[3],
and Yannis Velegrakis[4]

[1] ISI Foundation, Turin, Italy
`giulia.preti@isi.it`
[2] Amazon, Tokyo, Japan
`prrozens@amazon.co.jp`
[3] KTH Royal Institute of Technology, Stockholm, Sweden
`argioni@kth.se`
[4] University of Trento and Utrecht University, Utrecht, The Netherlands
`i.velegrakis@uu.nl`

Abstract. Given a dynamic network, where edges appear and disappear over time, we are interested in finding sets of edges that have similar temporal behavior and form a dense subgraph. Formally, we define the problem as the enumeration of the maximal subgraphs that satisfy specific density and similarity thresholds. To measure the similarity of the temporal behavior, we use the correlation between the binary time series that represent the activity of the edges. For the density, we study two variants based on the average degree. For these problem variants we enumerate the maximal subgraphs and compute a compact subset of subgraphs that have limited overlap. We propose an approximate algorithm that scales well with the size of the network, while achieving a high accuracy. We evaluate our framework on both real and synthetic datasets. The results of the synthetic data demonstrate the high accuracy of the approximation and show the scalability of the framework.

1 Introduction

A popular graph-mining task is discovering dense subgraphs, i.e., densely connected portions of the graph. Finding dense subgraphs was well studied in computer science and data-mining communities with many real-world applications.

Many highly dynamic real-life applications are modeled by continuously changing graphs. In some cases, the nodes and edges may evolve in a convergent manner and display correlated behavior. These groups of correlated elements, especially when they are topologically close, can represent regions of interest in the network. This work is focused on the discovery of such patterns, i.e., on the *correlated dense subgraphs* in dynamic networks. We consider graphs with edges that appear and disappear as time passes. Our goal is to identify sets of edges

---

that show a similar behavior in terms of their presence in the graph, and at the same time, are densely connected. Previous works [5,6] considered a similar problem, but limited to the time intervals and hard network partition. In this work we propose a general framework for finding dense correlated subgraphs, which can work with any temporal and spatial measure. Given specific density and correlation thresholds, we enumerate all maximal (the output set does not contain graphs, which are subgraphs of one another) subgraphs that satisfy the thresholds. Furthermore, since outputting a large number of highly overlapping subgraphs is not practical, we produce a manageable and informative set of highly diverse subgraphs.

Our main contributions are: **(i)** We introduce and formally define the generic problem of detecting a set of dense and correlated subgraphs in dynamic networks (Sect. 2), and explain how it differs from other similar works (Sect. 5); **(ii)** We propose two different measures to compute the density of a group of edges that change over time, which are based on the average-degree density [7], and a measure to compute their correlation, based on the Pearson correlation (Sect. 2); **(iii)** We develop an exact solution, called EXCODE, for enumerating all the subgraphs that satisfy given density and correlation thresholds. We also propose an approximate solution that scales well with the size of the network, and at the same time achieves high accuracy (Sect. 3); **(iv)** We study the problem of identifying a more compact and diverse subset of results. We extend our framework to extract a set of subgraphs with a pairwise overlap less than a specified threshold (Sect. 3); **(v)** We evaluate our framework on both real and synthetic datasets, confirming the correctness of the exact solution, the high accuracy of the approximate one, the scalability of the framework, and the applicability of the solution on networks of different nature (Sect. 4).

The extended version of the paper can be found at http://arxiv.org/abs/2103.00451.

2   Problem Statement

A *dynamic network* is a graph that models data that change over time. It is represented as a sequence of static graphs (*snapshots* of the network).

Dynamic Network. Let $T \subseteq \mathcal{T}$ be a set of time instances over a domain $\mathcal{T}$. A *dynamic network* $D = (V, E)$ is a sequence of graphs $G_i = (V, E_i)$ with $i \in T$, referred to as snapshots of the network, where V is a set of vertices, $E_i \subseteq V \times V$ is a set of edges between vertices. The set E denotes the union of the edges in the snapshots, i.e., $E = \cup_{i \in T} E_i$.

We assume that all the snapshots share the same set of nodes. If a node does not interact in a snapshot, then it is present as a singleton.

Given a graph $G = (V, E)$, a *subgraph* H of G is a graph $H = (V_H, E_H)$, such that $V_H \subseteq V$ and $E_H \subseteq E$. In static graphs, the density of a subgraph is traditionally computed as the average degree of its nodes [7]:

Density. The *density* of a (static) graph $G = (V, E)$ is the average degree of its nodes, i.e., $\rho(G) = 2|E|/|V|$.

In the case of a dynamic network D, the edges of a subgraph H may not exist in all the snapshots, meaning that the density may be different in each snapshot. Therefore, we propose two approaches to aggregate the density values. Let $G_i(H) = (V_H, E_H \cap E_i)$ denote the subgraph induced by H in the snapshot i. The *minimum density*, denoted as ρ_m, is the minimum density of any subgraph induced by H across the snapshots of D; while the *average density*, denoted as ρ_a, is the average density among these induced subgraphs. In particular,

$$\rho_m(H) = \min_{i \in T} \rho(G_i(H)), \quad \rho_a(H) = \frac{1}{|T|} \sum_{i \in T} \rho(G_i(H)). \tag{1}$$

Given a density threshold δ, a subgraph H is called δ-dense if $\rho_m(G) \geq \delta$ or $\rho_a(G) \geq \delta$, respectively.

These intuitive definitions are too strict for those practical situations where an interesting event or anomaly exhibits itself only in a small number of snapshots of the network [2]. To account for such situations, we introduce the notion of *activity* and say that a subgraph H is *active* at time t if at least k edges of H exist in t, i.e., $|E_t \cap E_H| \geq k$. Then, we relax our density definitions and compute the minimum and average density of H by aggregating only over the snapshots where H is active. Let T_H^k denote the subset of snapshots H is active, i.e., $T_H^k = \{t \mid t \in T \text{ and } |E_t \cap E_H| \geq k\}$. We redefine Eq. 1 as follows:

$$\rho_m^k(H) = \min_{i \in T_H^k} \rho(G_i(H)), \quad \rho_a^k(H) = \frac{1}{|T_H^k|} \sum_{i \in T_H^k} \rho(G_i(H)). \tag{2}$$

If T_H^k is empty, then both $\rho_m^k(H)$ and $\rho_a^k(H)$ are set to 0. We use the notation ρ^k to refer collectively at ρ_m^k and ρ_a^k.

We say that a subgraph is *correlated* if its edges are pairwise correlated. We therefore represent every edge as a time series over the snapshots, and measure the correlation between two edges as the Pearson correlation between the time series. Pearson correlation is widely used to detect associations between time series [8]. However, our framework can work with any other correlation measure. Let $\mathbf{t}(e)$ denote the time series of the edge e, where each coordinate is set to $t_i(e) = 1$ if e appears in the snapshot i, and thus $t_i(e) = 0$ otherwise.

Edge Correlation. Let $D = (V, E)$ be a dynamic network, $e_1, e_2 \in E$ be two edges with respective time series $\mathbf{t}(e_1) = \{t_1(e_1), \ldots, t_T(e_1)\}$, $\mathbf{t}(e_2) = \{t_1(e_2), \ldots, t_T(e_2)\}$, and $\bar{t}(e) = \frac{1}{|T|} \sum_{i=1}^{T} t_i(e)$. The correlation between e_1 and e_2, denoted as $c(e_1, e_2)$, is the Pearson correlation between $t(e_1)$ and $t(e_2)$, i.e.,

$$c(e_1, e_2) = \frac{\sum_{i=1}^{T} (t_i(e_1) - \bar{t}(e_1))(t_i(e_2) - \bar{t}(e_2))}{\sqrt{\sum_{i=1}^{T} (t_i(e_1) - \bar{t}(e_1))^2} \sqrt{\sum_{i=1}^{T} (t_i(e_2) - \bar{t}(e_2))^2}}.$$

Given a correlation threshold σ, the edges e_1 and e_2 are considered correlated if $c(e_1, e_2) \geq \sigma$.

We define the correlation of a subgraph H as the minimum pairwise correlation between its edges, i.e., $c_m(H) = \min_{e_i \neq e_j \in E_H} c(e_i, e_j)$, and say that H is σ-correlated if $c_m(H) \geq \sigma$.

Our goal is to identify all the dense and correlated subgraphs in a dynamic network. However, since a dense correlated subgraph may contain dense correlated substructures due to the nature of the density and correlation measures used, we restrict our attention to the *maximal* subgraphs. Thus, given a dynamic network D, a density threshold δ, and a correlation threshold σ, we want to find all the subgraphs H that are δ-dense and σ-correlated, and are not a strict subset of another δ-dense σ-correlated subgraph.

As it is often the case with problems that enumerate a complete set of solutions that satisfy given constraints, the answer set could potentially be very large and contain solutions with a large degree of overlap. To counter this effect, we further focus on reporting only the *diverse* subgraphs, which are subgraphs that differ from one another and are representative of the whole answer set. To measure the similarity between subgraphs, we use the Jaccard similarity between their edge sets, i.e., the Jaccard similarity between the graph $G'=(V', E')$ and $G''=(V'', E'')$, denoted as $J(G', G'')$, is $J(G', G'')=|E' \cap E''|/|E' \cup E''|$. Then, we require that the pairwise similarities between subgraphs in the answer set are lower than a given similarity threshold ϵ. This is in line with previous work that has aimed at finding a diverse collection of dense subgraphs [9].

Diverse Dense Correlated Subgraphs Problem [DICORDIS]. Given a dynamic network D, a density threshold δ, a correlation threshold σ, and a similarity threshold ϵ, find a collection $\mathcal{S}$ of maximal and diverse subgraphs such that for each $H \in \mathcal{S}$, H is δ-dense and σ-correlated, and for each distinct $H, H' \in \mathcal{S}$, $J(H, H') \leq \epsilon$.

3   Solution

To solve DICORDIS, we propose a two-step approach, called EXCODE (**Ex**tract **Co**rrelated **D**ense **E**dges). It first identifies maximal sets of correlated edges, and then extracts subsets of edges that form a dense subgraph according the density measures ρ_m^k or ρ_a^k. The correlation of a set of edges is computed as c_m.

Given the dynamic network $D = (V, E)$ we create a *correlation graph* $\mathcal{G} = (E, \mathcal{E})$, such that the vertex set of $\mathcal{G}$ is the edge set E of D, and the edges of $\mathcal{G}$ are the pairs $(e_1, e_2) \in E \times E$ that have correlation $c(e_1, e_2) \geq \sigma$. It is easy to see that a *maximal clique* in the correlation graph $\mathcal{G}$ corresponds to a maximal set of correlated edges in D.

The flow of EXCODE is illustrated in Algorithm 1. Starting from the dynamic network $D = (V, E)$ the algorithm first creates the correlation graph $\mathcal{G}$ by adding a meta-edge between two edges of D if their correlation is greater than σ. Then it enumerates all the maximal cliques in $\mathcal{G}$. This collection of maximal cliques in $\mathcal{G}$ corresponds to a collection $\mathcal{C}$ of maximal correlated edge sets in D. Finally, FINDDIVERSEDENSEEDGES examines each connected component in $\mathcal{C}$ (by using

---

Algorithm 1. EXCODE

Input: Dynamic network $D = (V, E)$, Density function ρ^k
Input: Thresholds: Correlation σ, Density δ, Size s_M
Input: Thresholds: Edges-per-snapshot k, Similarity ϵ
Output: Diverse dense correlated maximal subgraphs $\mathcal{S}$
 1: $\mathcal{G} \leftarrow$ CREATECORRELATIONGRAPH(G, σ)
 2: $\mathcal{C} \leftarrow$ FINDMAXIMALCLIQUES($\mathcal{G}$)
 3: $\mathcal{S} \leftarrow$ FINDDIVERSEDENSEEDGES($D, \mathcal{C}, \rho^k, \delta, k, s_M, \epsilon$)
 4: **return** $\mathcal{S}$

Algorithm 2. FINDDIVERSEDENSEEDGES

Input: Dynamic Network $D = (V, E)$
Input: Set of maximal cliques $\mathcal{C}$, Density function ρ^k
Input: Thresholds: Density δ, Size s_M, Edges-per-snapshot k, Similarity ϵ
Output: Set of diverse dense maximal subgraphs $\mathcal{S}$
 1: $\mathcal{S} \leftarrow \emptyset; \mathcal{P} \leftarrow \emptyset$
 2: $CC \leftarrow$ EXTRACTCC($\mathcal{C}$)
 3: **for each** $X \in CC$ **do**
 4:     **if** $X.size < s_M$ **and** ISMAXIMAL($X, \mathcal{S} \cup \mathcal{P}$) **and** ISDIVERSE($X, \mathcal{S}$) **then**
 5:         $(flag, R) \leftarrow$ ISDENSE(D, X, k, ρ^k, δ)
 6:         add X to $\mathcal{S}$ **if** $flag = 1$
 7:         add R to $\mathcal{P}$ **if** $flag = 0$
 8: **for each** $X \in \mathcal{P}$ **do**
 9:     add X to $\mathcal{S}$ **if** ISMAXIMAL($X, \mathcal{S}$) **and** ISDIVERSE(X, $\mathcal{S}$)
10: **return** $\mathcal{S}$

---

either the density measure ρ_m^k or ρ_a^k) to identify those constituting dense subgraphs in D, retaining only a subset of pairwise dissimilar subgraphs according to the similarity threshold ϵ. Next we describe the key elements of Algorithm 1, while the detailed descriptions of all the subroutines are in the supplementary materials.

Creation of the Correlation Graph. The correlation graph $\mathcal{G}$ can be built exactly, by computing the correlation $c(e_1, e_2)$ between each pair of edges $e_1, e_2 \in E$ and retaining those pairs satisfying $c(e_1, e_2) \geq \sigma$. However, when D is large, comparing each pair of edges is prohibitively expensive, and thus we propose an approximate solution based on *min-wise hashing* [3]. Here we exploit the fact that a strong correlation between two edges implies a high Jaccard similarity of the sets of snapshots where the edges appear. We use min-wise hashing to identify sets of candidate correlated edges. Specifically, we use a variant of the TAPER algorithm [19].

Enumeration of the Maximal Cliques. After the creation of the correlation graph $\mathcal{G}$, the maximal groups of correlated edges are enumerated by identifying the maximal cliques in $\mathcal{G}$. To this aim, we use our implementation of the *GP* algorithm of Wang et al. [17].

Algorithm 3. ISDENSE

Input: Dynamic Network $D = (V, E)$
Input: A set of edges X, Density function ρ^k
Input: Thresholds: Density δ, Edges-per-snapshot k
Output: $(1, \emptyset)$ if X is dense; $(0, R)$ if X contains the dense subsets R; $(-1, \emptyset)$ o.w.
 1: $K \leftarrow$ KEDGESNAPSHOTS(X, k)
 2: **if** $\rho^k(X, K, \delta)$ **then return** $(1, \emptyset)$
 3: **if** CONTAINSDENSE$(X, K, k) = \emptyset$ **then return** $(-1, \emptyset)$
 4: **return** $(0, $EXTRACTDENSE$(X, K, k))$

 5: **function** CONTAINSDENSE(X, K, k)
 6:     **while** $\rho^k(X, K, \delta/2) =$ **false do**
 7:         **if** $X = \emptyset$ **or** $K = \emptyset$ **then return false**
 8:         $max \leftarrow$ GETMAXDEG(X)
 9:         $n \leftarrow$ GETMINDEGNODE(X)
10:         **if** $max < \delta/2$ **then return false**
11:         $X \leftarrow X \setminus adj(n)$
12:         $K \leftarrow$ KEDGESNAPSHOTS(X, k)
13:     **return true**
14: **function** EXTRACTDENSE(X, K, k)
15:     $R \leftarrow \emptyset; Q \leftarrow \{(X, K)\}$
16:     **while** $Q \neq \emptyset$ **do**
17:         extract (Y, K') from Q
18:         **if** $\rho^k(Y, K', \delta)$ **then**
19:             $R \leftarrow R \cup \{Y\}$
20:         **else if** $K' \neq \emptyset$ **then**
21:             $max \leftarrow$ GETMAXDEG(Y)
22:             $N \leftarrow$ GETMINDEGNODES(Y)
23:             **if** $max < \delta$ **then continue**
24:             **for each** $n \in N$ **do**
25:                 $Y \leftarrow Y \setminus adj(n)$
26:                 $K' \leftarrow$ KEDGESNAPSHOTS(Y, k)
27:                 add (Y, K') to Q **if** $Y \neq \emptyset$
28:     **return** R

Discovery of the Dense Subgraphs. The goal of this step is to find con-
nected groups of edges that form a dense subgraph, using either ρ_m^k or ρ_a^k as
density function ρ^k. FINDDIVERSEDENSEEDGES receives in input a set of max-
imal cliques $\mathcal{C}$, each of which represents a maximal group of correlated edges.
Since some of the edges in a clique may not be connected in the network D,
the algorithm extracts all the distinct connected components from the cliques
(calling subroutine EXTRACTCC), before computing the density values. To allow
a faster discovery of the maximal groups of dense edges, the connected compo-
nents are sorted in descending order of their size and processed iteratively. If no
larger or similar dense set of the current candidate X has been discovered yet
(line 4), and if the size of X does not exceed the threshold s_M, the density of

X is computed by ISDENSE (line 5) calling subroutine ISAVGDENSE (for $\rho_a^k(X)$) or subroutine ISMINDENSE (for $\rho_m^k(X)$). Both subroutines iterate through the snapshots where at least k edges of X are present, but then the former aggregates the density values, while the latter keeps track of the minimum density. The latter allows for early stopping, if it encounters a snapshot where the density is below the threshold. However, thanks to the optimizations described in the next paragraph, the implementation of ISAVGDENSE is more efficient than that of ISMINDENSE, and thus we call the latter only when the former returns **true**, given that the average is an upper bound to the minimum.

When the density of the subgraph H induced by X is above the threshold δ, X is inserted in the result set $\mathcal{S}$ (line 6). Otherwise, some subset $X' \subseteq X$ may satisfy the condition $\rho_a^k(H') \geq \delta$. Since examining all the subsets of X is costly, we use Procedure CONTAINSDENSE, which is based on a 2-approximation algorithm for the densest subgraph problem [7], to prune the search space. In details, Procedure CONTAINSDENSE iteratively removes the vertex with lowest degree from the subgraph H, until it becomes empty or its density is greater than $\delta/2$. Every time a vertex is removed, its outgoing edges are removed as well (line 11), and thus the set of valid snapshots K must be updated (line 12). If K becomes empty, any subset of X has zero density, and thus the algorithm returns **false** (line 7). If the maximum value of density calculated during the execution of this algorithm is below the threshold $\delta/2$, it holds that X cannot contain a subset X' with density above δ [7], and thus CONTAINSDENSE returns **false**. Therefore, EXTRACTDENSE, which extracts all the dense subsets in the set X, is invoked (line 4) only when CONTAINSDENSE returns **true**.

When Procedure CONTAINSDENSE returns **true**, EXTRACTDENSE iteratively searches for all the dense subsets in X. At each iteration, a subset of edges Y is extracted from the queue Q and its density is checked. If Y is not dense but the set of valid snapshots is not empty (line 20), a new candidate is created for each vertex n with lowest degree in the subgraph induced by Y. These candidates are then inserted into Q. On the other hand, when Y is dense, it is inserted into the result set R. At the end of Algorithm 2, the maximal subsets in the set $\mathcal{P}$, which contains the elements of all the R sets computed during the search, are checked for similarity with the subsets already in $\mathcal{S}$. Those with Jaccard similarity below ϵ with any subsets in $\mathcal{S}$ are finally added to $\mathcal{S}$ (lines 8–9).

Computing Average Density Efficiently. The average density $\rho_a(H)$ of a subnetwork $H = (V_H, E_H)$ in a dynamic network D can be computed via the *summary graph* of D defined as the *static graph* $\mathcal{R} = (V, E, \sigma)$ where V is the set of vertices of D, E is the union of the edges E_i of all the snapshots of D, and $\sigma : E \mapsto \mathbb{R}$ is a weighting function that assigns, to each edge $e \in E$, a value equal to its average appearance over all the snapshots of D, i.e., $\sigma(e) = 1/|T| \sum_{i \in T} t_i(e)$. The following proposition ensures that $\rho_a(H)$ is equivalent to the weighted density of H in the summary graph $\mathcal{R}$, which is defined as $w\rho(H) = 2 \sum_{e \in E_H} \sigma(e)/|V_H|$.

Proposition 1. *Given a dynamic network D, its summary graph $\mathcal{R}$, and a subnetwork H, it holds that $\rho_a(H) = w\rho(H)$.*

Proof.

$$\rho_a(H) = \frac{1}{|T|} \sum_{t \in T} \rho(G_t(H)) = \frac{1}{|T|} \sum_{i \in T} \left(\frac{2|E_H \cap E_t|}{|V_H|} \right)$$

$$= \frac{2}{|V_H|} \frac{1}{|T|} \sum_{i \in T} \sum_{e \in E_H} t_i(e) = \frac{2}{|V_H|} \sum_{e \in E_H} \sigma(e) = w\rho(H). \quad \square$$

The weighted density of H in the summary graph $\mathcal{R}$ can be calculated significantly faster than its average density in the dynamic network D, since the former is obtained by summing the appearances of the edges of H defined by σ, while the latter is obtained by constructing the subgraph induced by E_H in each snapshot, computing the average node degree of each induced subgraph, and taking the average among those values. Thus, Proposition 1 allows us to improve the efficiency of our algorithm when using ρ_a (and ρ_a^k) density function.

ExCoDe Complexity. The exact construction of $\mathcal{G}$ takes $\mathcal{O}(|E|^2)$, as it requires the computation of all the pairwise edge correlations. The approximate solution creates $h \cdot r$ hash values for the edges in $\mathcal{O}(h \cdot r \cdot |E|)$ and compares only the edges that share at least one hash code. Even though the worst-case time complexity is still $\mathcal{O}(|E|^2)$ (every pair of edges share some hash code), practically, the actual number of comparisons is much smaller than $|E|^2$. The time complexity of the maximal clique enumeration is $\mathcal{O}(|E| \cdot \kappa(\mathcal{G}))$, where $\kappa(\mathcal{G})$ is the number of cliques in $\mathcal{G}$. The computation of the connected components in the maximal cliques takes $\mathcal{O}(|E| \cdot \kappa(\mathcal{G}))$, as it requires a visit of the network D for each clique. In the worst case, each edge of the network belongs to a different connected component, and thus Algorithm 2 must iterate $|E|$ times. At each iteration, it calls Procedure ISDENSE to compute the density of the current set of edges X if its size is lower than s_M. Procedure ISDENSE calculates the average node degree of each subgraph induced by X in all the snapshots where at least k edges of X are present (at most $|T|$), and thus its time is bounded by $\mathcal{O}(s_M \cdot |T|)$. When X is not dense, the algorithm further calls Procedure CONTAINSDENSE and Procedure EXTRACTDENSE. The former runs in s_M, since it removes at least one edge from X at each iteration; while the latter must process all the subsets of X in the worst case (2^{s_M}). The complexity of Algorithm 2 is therefore $\mathcal{O}(\kappa(\mathcal{G}) \cdot |E| + |E|(s_M \cdot |T| + s_M + 2^{s_M})) = \mathcal{O}(|E|(\kappa(\mathcal{G}) + s_M|T| + 2^{s_M}))$, which is also the complexity of Algorithm 1.

4   Experimental Evaluation

We evaluate the performance of our exact and approximate solutions in terms of accuracy and execution time. We also integrated our solution into a tool demonstrated at ICDMW19 [14]. More experimental results can be found in the supplementary materials, due to space limitations.

The datasets considered are 3 real networks and 6 randomly-generated networks, the characteristics of which are shown in Tables 1 and 2, respectively. They report the number of vertices $|V|$, edges $|E|$, and snapshots $|T|$; the average node degree $d_a(G)$; the average node degree per snapshot $d_a(G_i)$; and the

Table 1. Real datasets

Dataset	$|V|$	$|E|$	$|\mathcal{T}|$	$d_a(G)$	$d_a(G_i)$	$c_a(e)$
HAGGLE	274	2K	90	15.5	5.2	5.4
TWITTER-S	767	2K	2K	6.2	3	121.1
TWITTER-M	1.2K	7K	2K	12.1	3.2	86.4
TWITTER-L	1.3K	10K	2K	15.2	3.3	68.6
MOBILE-S	5K	42K	48	15.3	4.9	3.8
MOBILE-M	5K	80K	48	28.6	6.4	3.6
MOBILE-L	5K	118K	48	41.4	7.5	3.6

Table 2. Synthetic datasets

Dataset	$|V|$	$|E|$	$|\mathcal{T}|$	p_{in}	p_{out}	Independent			Correlated	
						$d_a(G)$	$d_a(G_i)$	$a_a(e)$	$d_a(G_i)$	$c_a(e)$
GAUSSIAN-1-7-1	100	1059	100	0.7	0.1	21.1	10.6	50	10.32	48.2
GAUSSIAN-2-7-1	200	3029	100	0.7	0.1	30.2	15.1	50.1	15.1	50.1
GAUSSIAN-3-7-1	300	6070	100	0.7	0.1	40.4	20.2	50	20.3	50.3
GAUSSIAN-1-7-3	100	1825	100	0.7	0.3	36.5	18.2	50	18.2	49.9
GAUSSIAN-2-7-3	200	6828	100	0.7	0.3	68.2	34	49.9	33.8	49.6
GAUSSIAN-3-7-3	300	14723	100	0.7	0.3	98.1	49	49.9	48.9	49.8

Table 3. Minimum (F_m) and average (F_a) F-score, running time. Worst case values in parenthesis. "-" for runs longer than 2 days.

Dataset	CIFORAGER			EXCODE		
	F_m	F_a	$t(min)$	F_m	F_a	$t(sec)$
GAUSSIAN-1-7-1	0	(.07) .03	(19) .2	(.98) 1	(.99) 1	(1.1) .71
GAUSSIAN-2-7-1	0	(.03) .01	(365) 4	(.00) 1	(.95) 1	(2.2) 1.6
GAUSSIAN-3-7-1	0	(-) .007	(-) 44	(.98) 1	(.99) 1	(8.0) 4.5
GAUSSIAN-1-7-3	0	(.02) .01	(78) .9	(.98) 1	(.99) 1	(1.2) 1
GAUSSIAN-2-7-3	0	(-) .003	(-) 332	(.97) 1	(.99) 1	(10) 5.5
GAUSSIAN-3-7-3	-	(-) -	(-) -	(.98) 1	(.99) 1	(51) 23

average number of appearances of an edge in the snapshots $c_a(e)$. HAGGLE [4] is a human-contact network, TWITTER [12] is a hashtag co-occurrence network created using tweets collected from 2011 to 2016, and MOBILE [11] is a network modeling calls between users made available by Telecom Italia. The GAUSSIAN-X-Y-Z are synthetic networks generated using the *gaussian random partition graph* generator in the Python NetworkX library[1]. A graph is obtained by partitioning
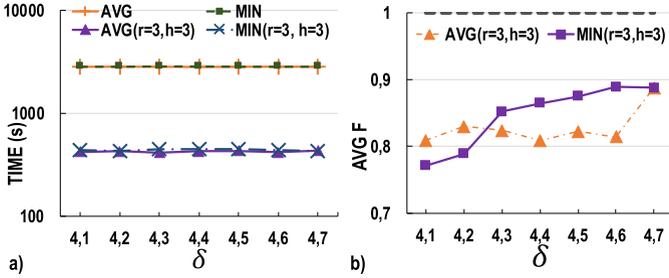
---

[1] https://tinyurl.com/y5sezq73.

Fig. 1. Running time and average F score for varying δ

the set of n nodes into k groups each of size drawn from a normal distribution $\mathcal{N}(s, s/v)$, and then adding intra-cluster edges with probability p_{in} and inter-cluster edges with probability p_{out}.

We implemented our algorithms in Java 1.8, and run the experiments on a 24-Core (2.40 GHz) Intel Xeon E5-2440 with 188 Gb RAM with Linux 3.13, limiting the amount of memory available to 150 Gb. In addition, we implemented a Java version of CIFORAGER [6], which is the approach most related to ours. For the synthetic datasets we report results based on 100 runs.

Effectiveness of the Exact Solution. We tested the effectiveness of our exact algorithm in detecting the actual dense groups of correlated edges in the synthetic networks GAUSSIAN-X-7-1 and GAUSSIAN-X-7-3. The correlation threshold σ is set to 0.8; the density threshold δ is equal to the minimum average degree among the actual dense groups, namely 2; and the maximum size s_M is set to ∞ to ensure we do not miss any dense group. We measured the accuracy in terms of the Jaccard similarity between the groups of edges discovered $\mathcal{S}$ and the dense groups in the ground-truth $\mathcal{G}$. First, for each group $H \in \mathcal{S}$, we computed the Jaccard similarity with its closest dense group in $\mathcal{G}$, and then calculated minimum and average precision as P_a and P_m:

$$P_a = \frac{1}{|\mathcal{S}|} \sum_{H \in \mathcal{S}} \max_{J \in \mathcal{G}} \text{JACC}(H, J) \qquad P_m = \min_{H \in \mathcal{S}} \max_{J \in \mathcal{G}} \text{JACC}(H, J)$$

Then, for each dense group $J \in \mathcal{G}$, we computed the Jaccard similarity with its closest group in $\mathcal{S}$, and calculated minimum and average recall R_a and R_m:

$$R_a = \frac{1}{|\mathcal{G}|} \sum_{J \in \mathcal{G}} \max_{H \in \mathcal{S}} \text{JACC}(H, J) \qquad R_m = \min_{J \in \mathcal{G}} \max_{H \in \mathcal{S}} \text{JACC}(H, J)$$

Finally, we report the average and minimum F-score, as: $F_a = 2(P_a \cdot R_a)/(P_a + R_a)$ and $F_m = 2(P_m \cdot R_m)(P_m + R_m)$.

As shown in Table 3, for each synthetic network EXCODE obtained both $F_a = 1$ and $F_m = 1$, meaning that the algorithm correctly identified all the dense groups despite the extra edges added between the groups in the networks. We achieved lower scores only when using correlation thresholds $\sigma \leq 0.2$ for the smallest network, and $\sigma \leq 0.3$ for the others. In these cases it is more likely

that some inter-group edges have a correlation greater than σ, and therefore the algorithm discovers sets of edges that are supersets of the actual dense groups. Nonetheless, the F_a score is always greater than 0.94, while the F_m score is lower than 0.97 only for network GAUSSIAN-2-7-1.

In addition, Table 3 compares our approach with CIFORAGER [6], the closest competitor. CIFORAGER creates a partition of edges to optimize temporal (aggregated over temporal windows) correlation and spatial (all-pairs path) distance of each partition. We run the code with the default parameters: window length $w_l = 10$, window overlap size $w_i = 1$, clustering similarity threshold 0.25, region similarity threshold 0.2. The window length is the size of the window used to segment the sequence of graph snapshots into overlapping subsequences, while the overlap size indicates how much the subsequences overlap. The clustering threshold decides which edges to be grouped together, and the region threshold determines how to merge the groups of edges found in different windows.

Since the output of CIFORAGER is an edge partition, and thus contains edges that are not part of any dense group, the average and minimum precisions P_a and P_m are always low, and, as a consequence, the F_a and F_m are always lower than those of EXCODE. Performance wise, CIFORAGER is expensive since it computes the temporal distance for almost each pair of edges and for each window. In contrast, our algorithm was able to terminate in less than a minute with every configuration and network tested.

Efficiency of the Approximate Solution. Figure 1 shows the performance and running time of EXCODE *approximate* to find the (0.9)-correlated δ-dense subgraphs in the MOBILE-M network, using both ρ_a^k and ρ_m^k, and varying δ. As we can see, the approximate solution is one order of magnitude faster than the exact algorithm, and yet achieves a F_a score of at least 0.8 for the ρ_a^k (AVG) case, and 0.77 for the ρ_m^k (MIN). We observed a similar behavior also in the other networks. As an example, in the TWITTER samples we obtained the highest F_a score at high density values, while a minimum of 0.63 at low density values.

5   Related Work

Our problem is close to dense subgraph mining in dynamic networks. Works in this field aim at retrieving the highest-scoring temporal subgraph [13], the densest temporally compact subgraph [15], or the group of nodes most densely connected in all the snapshots [16]. Although they can be adapted to retrieve multiple subgraphs, the detected subgraphs are non-overlapping, and with edges that are not temporally correlated. The enumeration of dense structures has been studied in the context of frequent subgraph mining [1], and top densest subgraph mining [9]. When the input is a dynamic network, these groups represent subgraphs that persist over time; however, in general they are not temporally correlated. In anomaly and fraud detection, other measures have been considered, together with the density, with the goal of finding interesting regions in the snapshots of a dynamic network [2]. All such works focus on the statistically significant structures, while our interest is on dense groups of edges with a

similar behavior over time. A notion of correlation has been used in approaches that characterize the event dynamics by the number of articular labels in the vicinity of spacial reference nodes [10], or that compute a decay factor [18]. In contrast to our work, both these approaches retrieve only anomalous nodes. The works closest to ours are CStag [5] and its incremental version ciForager [6], which find regions of correlated temporal change in dynamic graphs. However, they partition the edges into L regions, meaning that each edge is a part of the output, and hence the output can be very large and contain a number of low quality graphs. In contrast, we enumerate only the subgraphs with large density and high pairwise edge correlation.

6   Conclusions

We studied the problem of finding maximal dense correlated subgraphs in dynamic networks. We proposed two measures to compute the density of a subgraph that changes over time, and one to assess the temporal correlation of its edges. We described a framework that uses those measures to identify such subgraphs for given density and correlation thresholds. We experimentally demonstrated the limitations of the existing solutions and provided an approximate solution that runs in an order of magnitude faster, yet achieving a good solution quality.

References

1. Abdelhamid, E., Canim, M., Sadoghi, M., Bhattacharjee, B., Chang, Y.C., Kalnis, P.: Incremental frequent subgraph mining on large evolving graphs. TKDE **29**(12), 2710–2723 (2017)
2. Akoglu, L., Tong, H., Koutra, D.: Graph based anomaly detection and description: a survey. Data Min. Knowl Disc. **29**(3), 626–688 (2014). https://doi.org/10.1007/s10618-014-0365-y
3. Broder, A.Z., Charikar, M., Frieze, A.M., Mitzenmacher, M.: Min-wise independent permutations. JCSS **60**(3), 630–659 (2000)
4. Chaintreau, A., Hui, P., Crowcroft, J., Diot, C., Gass, R., Scott, J.: Impact of human mobility on opportunistic forwarding algorithms. Trans. Mob. Comput. **6**(6), 606–620 (2007)
5. Chan, J., Bailey, J., Leckie, C.: Discovering correlated spatio-temporal changes in evolving graphs. KAIS **16**(1), 53–96 (2008)
6. Chan, J., Bailey, J., Leckie, C., Houle, M.: ciForager: incrementally discovering regions of correlated change in evolving graphs. TKDD (2012)
7. Charikar, M.: Greedy approximation algorithms for finding dense components in a graph. In: APPROX (2000)

8. Fu, T.C.: A review on time series data mining. Eng. Appl. Artif. Intell. **24**(1), 164–181 (2011)
9. Galbrun, E., Gionis, A., Tatti, N.: Top-k overlapping densest subgraphs. Data Min. Knowl. Disc. **30**(5), 1134–1165 (2016)
10. Guan, Z., Yan, X., Kaplan, L.M.: Measuring two-event structural correlations on graphs. PVLDB **5**(11) (2012)
11. Italia, T.: Telecommunications - tn to tn (2015). https://doi.org/10.7910/DVN/KCRS61
12. Lahoti, P., Garimella, K., Gionis, A.: Joint non-negative matrix factorization for learning ideological leaning on twitter. In: WSDM (2018)
13. Ma, S., Hu, R., Wang, L., Lin, X., Huai, J.: Fast computation of dense temporal subgraphs. In: ICDE (2017)
14. Preti, G., Rozenshtein, P., Gionis, A., Velegrakis, Y.: Excode: a tool for discovering and visualizing regions of correlation in dynamic networks. In: ICDMW (2019)
15. Rozenshtein, P., Tatti, N., Gionis, A.: Finding dynamic dense subgraphs. TKDD **11**(3), 1–30 (2017)
16. Semertzidis, K., Pitoura, E., Terzi, E., Tsaparas, P.: Finding lasting dense subgraphs. Data Min. Knowl. Disc. **33**(5), 1417–1445 (2018). https://doi.org/10.1007/s10618-018-0602-x
17. Wang, Z., et al.: Parallelizing maximal clique and k-plex enumeration over graph data. J. Parallel Distr. Comput. **106**, 79–91 (2017)
18. Yu, W., Aggarwal, C.C., Ma, S., Wang, H.: On anomalous hotspot discovery in graph streams. In: ICDM (2013)
19. Zhang, J., Feigenbaum, J.: Finding highly correlated pairs efficiently with powerful pruning. In: CIKM (2006)